INTRODUZIONE

Questa tesi illustra l'intero svolgimento di un progetto studiato per sviluppare una soluzione dalle caratteristiche innovative ed il cui scopo sia supportare l'attività di monitoraggio di prezzi e tariffe. Il progetto prevede la realizzazione di un'architettura *On-Line Analytical Processing* capace di gestire un'elevata mole di dati e processare analisi avanzate.

Il monitoraggio di prezzi e tariffe ha da sempre rappresentato un'attività di vitale importanza per sostenere la trasparenza e promuovere la conoscenza sui livelli e sulle dinamiche che caratterizzano l'andamento dei prezzi, in tale ottica il progetto affrontato mira alla definizione di un evoluto strumento di analisi a supporto degli operatori economici e dei consumatori.

L'attività progettuale è nata da una preliminare fase di raccolta dei requisiti in cui sono stati analizzati gli osservatori prezzi online attualmente disponibili; la documentazione ricavata ha consentito di studiare le caratteristiche dei servizi attuali mettendo in evidenza limiti e carenze. L'analisi dei requisiti ha poi portato alla scelta di una direzione di progetto con l'obiettivo di sviluppare un'architettura che si distingua per funzionalità e capacità analitica rispetto ai servizi esaminati.

Il web offre un vasto patrimonio informativo cui attingere, spesso però le informazioni sono presentate all'utente a livello di contenuto testuale e non vi è un reale supporto all'accesso e all'eventuale integrazione delle sorgenti dati. Da tale considerazione è nata l'idea di realizzare un sistema, mirato al monitoraggio dei prezzi, che sia capace di analizzare i contenuti delle diverse pagine web ed automatizzare un processo di estrazione, interpretazione e caricamento dei dati verso una banca dati opportunamente strutturata.

Si è realizzata una vera e propria architettura *OLAP*, indirizzata verso analisi di alto livello, la cui caratteristica di rilievo è rappresentata dalla possibilità di recuperare le informazioni, per alimentare la base dati, direttamente dalle sorgenti web; ciò consente prima di tutto di abbattere i costi che sarebbero necessari per sostenere l'attività di rilevamento dei dati ed inoltre permette di integrare informazioni provenienti da più fonti verso un unico database. Il fine è di riuscire ad amministrare la continua proliferazione delle fonti informative e di mettere a disposizione dell'utente uno strumento di analisi ad alto valore aggiunto.

Le architetture *OLAP* si basano su *data warehouse*, ovvero su magazzini di dati opportunamente strutturati per sostenere le attività analitiche; lo scopo è di definire un modello che consenta di effettuare analisi dei dati su strutture multidimensionali in maniera rapida, flessibile ed efficiente, attraverso i servizi forniti da motori di database specifici.

Le analisi multidimensionali consistono nel "navigare" i dati lasciando all'utente la facoltà di scegliere interattivamente le informazioni da visualizzare ed i filtri da applicare; un sistema che sia in grado di supportare analisi di questo genere in riferimento a prezzi e tariffe rappresenta indubbiamente uno strumento di consistente validità.

Il progetto verrà discusso in tutte le sue fasi, a partire dalla raccolta dei requisiti fino alla sperimentazione, motivando ogni valutazione fatta e scelta effettuata. Si dedicherà molta attenzione all'attività di progettazione ed implementazione del data warehouse ed alla realizzazione di un modulo *ETL* (Extract, Transform e Load) costruito ad hoc per interpretare le informazioni estratte direttamente dalle pagine web. Ampio spazio verrà poi riservato alla scelta ed alla configurazione di un *OLAP Server* in grado di soddisfare le esigenze di analisi richieste.

L'obiettivo finale è di mettere in piedi un sistema capace di esplorare le informazioni disponibili sul web ed automatizzare un processo di estrazione, trasformazione e caricamento per alimentare il data warehouse modellato. L'attività progettuale si concentrerà quindi anche sull'analisi dei servizi web per il monitoraggio dei prezzi attualmente disponibili, cercando di individuare delle fonti che siano aggiornate, ricche ed affidabili; il sistema si occuperà poi di integrare in modo omogeneo i flussi informativi più appropriati per costituire un patrimonio informativo globale.

Nei successivi capitoli si riporterà un'esposizione dettagliata di tutte le fasi che hanno costituito il progetto, partendo dello studio dell'ambito applicativo per poi passare alla descrizione delle idee progettuali ed alla realizzazione dell'architettura pianificata.

Nel capitolo 1 si descriverà l'attività di raccolta dei requisiti svolta, si esporrà la metodologia adottata per analizzare gli osservatori prezzi online per poi trarre le dovute considerazioni sulle caratteristiche dei principali servizi.

Nel secondo capitolo si approfondirà l'analisi dei requisiti per stilare una serie di idee progettuali da poter sviluppare; verranno poi esposte le motivazioni che hanno indotto a scegliere una specifica direzione di progetto.

Nel capitolo 3 verrà descritta, dal punto di vista progettuale, l'architettura che è stata sviluppata; si approfondirà il ruolo di ogni singolo componente del sistema andando ad illustrare le problematiche da affrontare e le tecniche di progettazione adottate per il data warehouse.

Nel quarto capitolo si darà ampio spazio alla descrizione dalla fase implementativa. Si descriveranno gli algoritmi, le logiche, i linguaggi e gli strumenti utilizzati per implementare ogni modulo dell'architettura *OLAP*; si parlerà dettagliatamente delle specifiche e delle condizioni di sviluppo dell'intero applicativo.

Infine nel capitolo 5 si riporteranno gli esiti sperimentali ottenuti in seguito ad un periodo di test; verrà esplicitata la metodologia impiegata per condurre la sperimentazione e verranno espresse le opportune valutazioni sui risultati raggiunti.

4.5.3 SCELTA OLAP SERVER

In fase di implementazione è stata posta grande attenzione sulla scelta degli strumenti da utilizzare per la realizzazione dell'architettura. Particolare interesse ha richiesto la fase di scelta del Data Warehouse Server; l'intera architettura è orientata verso applicativi open-source e quindi anche la scelta dello strumento di supporto analitico ha seguito la direzione della "filosofia open" e si è focalizzata su un insieme ben definito di soluzioni già esistenti ed utilizzabili.

Si è quindi intrapreso uno studio sui prodotti di Business Intelligence open-source per riuscire, dopo un'attenta valutazione, ad individuare l'applicativo migliore da sfruttare per compiere analisi sui dati estratti. Così, mentre per l'attività di ETL si è deciso di progettare una soluzione personalizzata strettamente modellata intorno all'ambito applicativo di riferimento, per la gestione del sistema OLAP si è indirizzata la scelta verso soluzioni *general purpose* con potenzialità tali da riuscire ad adattare ogni funzionalità in riferimento all'ambito applicativo considerato.

In linea di massima i criteri di valutazione adottati, per individuare il prodotto migliore da impiegare, sono stati i seguenti:

- *costi*
- *flessibilità*
- *funzionalità*
- *interfaccia grafica*
- *requisiti minimi*
- *semplicità di utilizzo*

Costi

Uno dei principali aspetti da valutare nella scelta del software da utilizzare riguarda l'impiego di risorse in termini di costi.

Gran parte degli strumenti OLAP sono software proprietari piuttosto elaborati che offrono sia la piattaforma per la realizzazione del database, sia gli strumenti necessari per l'analisi dei dati.Molte software house propongono Client appositamente realizzati per interfacciarsi con i database Server e fornire duttili soluzioni di reporting ed utilità di analisi sui dati.

Il problema principale per questo genere di prodotti è che rappresentano un notevole investimento in termini di costi. D'altra parte però è possibile concentrare la scelta dell'applicativo verso soluzioni free, trial o open-source.

Per l'architettura in esame si è scelto di ridurre al minimo l'impiego di costi per il software ricorrendo ad applicazioni open-source. Naturalmente questa scelta rappresenta un restrizione dell'universo delle soluzioni esistenti e quindi la procedura di selezione dell'applicativo si basa su un insieme limitato di prodotti e deve necessariamente essere fatta con molta cura.

Flessibilità

La flessibilità rappresenta una proprietà decisamente importante nella scelta di un prodotto. In ambito applicativo il concetto di flessibilità esprime la capacità di un sistema di essere personalizzato ed adattato in riferimento a problemi di studio differenti.

Un software flessibile consente di effettuare modifiche ed impostazioni sulla base delle proprie esigenze non snaturando la funzionalità dell'applicativo ma adattandola a diversi casi di studio.

In riferimento a sistemi OLAP un software di gestione flessibile deve offrire la possibilità di definire ed amministrare il proprio data warehouse consentendo di sviluppare filtri, interrogazioni, report ed estrazioni di dati secondo le proprie necessità.

Funzionalità

La funzionalità di un prodotto software misura le opportunità di elaborazione che l'applicativo fornisce. Per un sistema di gestione OLAP il grado di funzionalità può essere espresso stabilendo l'ampiezza delle possibilità di elaborazione che il prodotto mette a disposizione. Un applicativo ricco di funzionalità ha come obiettivo quello di fornire numerose opportunità di trattamento e lavorazione dei dati; oltre agli strumenti standard di amministrazione il software può acquisire maggiori funzionalità basandosi sull'*interfacciabilità* con sistemi esterni.

Un'ulteriore sezione in cui è possibile quantificare le funzionalità di una applicazione è stabilita dalle proprietà grafiche che il prodotto mette a disposizione dell'utente. Le funzionalità grafiche di un sistema di gestione si esprimono esaminando le possibilità di rappresentare dati e risultati, ottenuti in seguito ad operazioni di analisi sui dati, in forma grafica (creazione di grafici, istogrammi, etc.).

Interfaccia Grafica

Molto spesso l'interfaccia grafica ha notevole importanza nella scelta di un prodotto software. Una buona *interfaccia grafica* deve mettere l'utente che utilizza l'applicazione in condizioni di chiarezza rispetto alle operazioni da eseguire.

La valutazione dell'interfaccia grafica per l'utilizzazione di un applicativo pone come riferimento la stima della *semplicità di utilizzo* del programma.

In riferimento ai sistemi OLAP le proprietà grafiche sono esprimibili analizzando le interfacce grafiche fornite con i Client che interagiscono con il Data Warehouse Server.

Requisiti Minimi

I requisiti minimi richiesti si valutano stabilendo le dipendenze del programma. Un programma per funzionare correttamente molto spesso richiede la presenza o l'installazione di specifici componenti software; questo aspetto rappresenta generalmente un vincolo in fase di scelta.

Molti software OLAP richiedono la presenza di specifiche piattaforme o librerie per lavorare regolarmente o necessitano di interagire con determinati database; in questo senso dei requisiti minimi troppo restrittivi possono indurre l'utente ad optare per una soluzione alternativa.

Semplicita' Di Utilizzo

Il livello di semplicità nell'utilizzo di un software si esprime giudicando le difficoltà di impiego dell'applicazione nella gestione delle operazioni che deve implementare.

Il concetto di semplicità per software gestionali è strettamente correlato al concetto di interfaccia grafica. Generalmente le *regole di interazione* tra programma ed utente vengono sviluppate in riferimento alla classe di utenti a cui l'applicazione è destinata.

Se il software è destinato ad utenti con poche competenze informatiche, le azioni di interazione devono essere sviluppate in modo da rendere la comunicazione con il programma più chiara possibile: l'interfaccia grafica deve essere comprensibile e limpida.

Riassumendo, uno strumento software può essere ritenuto semplice dal punto di vista del suo utilizzo se rispetta principalmente queste caratteristiche:

- Immediatezza nell'esecuzione delle operazioni
- "Navigabilità" trasversale per il recupero dell'informazione
- Grafica intuitiva

La semplicità può essere valutata anche dal punto di vista tecnico, classificando le applicazioni sulla base della loro *semplicità di installazione e configurazione*.

Per quanto riguarda gli strumenti informatici OLAP valutare la semplicità significa analizzare le difficoltà nell'installazione e nella configurazione del sistema di gestione, nonché giudicare l'interfaccia grafica utilizzata per interagire con il server OLAP.

Aspetti da osservare:

- Semplicità di installazione e configurazione del sistema
- Semplicità di accesso ed esecuzione alle funzionalità fornite dal sistema
- Semplicità nella creazione ed esecuzione di Query, Filtri e Report

Soluzioni open-source - motivazioni:

Nei confronti dell'adozione di software open source, l'atteggiamento è spesso ambivalente: da un lato, la prospettiva dell'immediato vantaggio economico dato dall'abbattimento dei costi delle licenze la rende attraente; dall'altro, l'assenza di un partner commerciale di riferimento la fa percepire come una soluzione "insicura" rispetto a quelle fornite dai canali tradizionali.

La situazione reale è sfaccettata: a settori in cui l'open source è storicamente forte (per citarne alcuni: web server, mail server, firewall) se ne stanno affiancando altri, come ad esempio gestione dei contenuti, archiviazione documenti, database, in cui si assiste ad una decisa discesa in campo dei maggiori *vendor* che stringono alleanze con consolidati progetti open source o contribuiscono liberalizzando il loro codice sorgente.

Vantaggi dell'open source:

- Informazioni e protocolli di comunicazione non sono in formati proprietario
- Molteplicità di strumenti (lo stesso problema può essere risolto scegliendo fra software alternativi)
- Elevata scalabilità (dipende in parte dalle architetture flessibili adottate da molti progetti open source, in parte dalla sostituibilità di uno strumento con altri con performance più adeguate)
- Forte adattabilità alle esigenze (uno strumento open source solitamente non è un prodotto finito monolitico, ma è altamente configurabile)
- Vantaggio economico (immediato, derivante dall'assenza di costi di licenze, ed indiretto, derivante dalle caratteristiche del software)

Punti critici:

- Scelta del settore applicativo: va fatta accuratamente, in base ad alcuni criteri euristici: settore ben conosciuto in ambito aziendale ed a basso impatto, possibilità di un coinvolgimento progressivo degli utenti, scelta di strumenti di sviluppo affidabili.
- Valutazione del progetto open source: le componenti della soluzione vanno scelte in base a: capacità del progetto, facilità di integrazione, sostituibilità, consistenza del team di progetto , qualità del processo di sviluppo, possibilità di auto-supporto, supporto della comunità.

Software testati

Gli strumenti presi in considerazione rappresentano alcuni dei prodotti di Business Intelligence open-source maggiormente affermati nel panorama delle soluzioni software attualmente esistenti.

La valutazione di ogni prodotto è stata sviluppata facendo riferimento ad ognuno dei criteri prefissati; sono stati esaminati diverse classi di prodotti, dalle soluzioni con funzionalità di semplice reportistica fino a soluzioni più sofisticate capaci di sviluppare un potenziale analitico più elevato.

Enterprise Reporting

Gli strumenti di Reporting forniscono un modo insostituibile di veicolare informazioni mirate, volte a supportare il monitoraggio delle performance aziendali e le attività decisionali. Le caratteristiche principali sono:

- integrazione nella piattaforma aziendale
- realizzazione di web application
- personalizzazione
- disegno autonomo di templates

Esistono molte soluzioni per le applicazioni di reportistica aziendale, tra esse due progetti tra i più rappresentativi sono il progetto *JasperReports* ed il progetto *Eclipse BIRT*.

Ad una robusta base di features fondamentali, ciascuno dei due progetti aggiunge delle caratteristiche peculiari che lo rendono adatto ad un'integrazione in ambito applicativo piuttosto che alla pubblicazione di reports aziendali su web, fermo restando la possibilità di entrambe di coesistere nella stessa piattaforma che ospita strumenti analitici open source, fornendo così una soluzione completa[33].

JasperReports e Birt rappresentano strumenti strettamente rivolti alla reportistica, adatti a disseminare informazioni verso moltitudini di utenti occasionali o decision-maker di alto livello. Entrambi strumenti potenti e flessibili, il primo più orientato all'integrazione con applicazioni stand-alone, il secondo più adatto per web application. Indipendenti dalle sorgenti dati e dalla piattaforma.

Decision Support System

I DSS offrono strumenti che agevolano i processi decisionali, consentendo all'utilizzatore di estrarre, elaborare, combinare ed analizzare le numerose informazioni; gli applicativi per il supporto alle decisioni implementano differenti funzionalità di analisi: nelle fasi iniziali esse riguardano principalmente la *reportistica* per poi svilupparsi nell'affermazione di *query* più elaborate fino a giungere a funzionalità OLAP *(On-Line Analystic Processing).*

In riferimento a questa classe di soluzioni si sono valutate applicazioni di analisi dati e supporto alle decisioni web-based che risultino altamente scalabili e facilmente personalizzabili.

In questa categoria di applicazioni una delle soluzioni più rilevanti è rappresentata dal software realizzato dai progetti *Mondrian* e *JPivot*. Si tratti di soluzioni analitiche di alto

livello ed ampiamente sviluppate; Mondrian rappresenta un vero e proprio motore OLAP attraverso il quale è possibile processare query multidimensionali in riferimento al proprio data warehouse; JPivot è, invece, un modulo che sfrutta Mondrian per gestire una tabella OLAP su cui è possibile generare dinamicamente delle operazioni di filtraggio sui i dati e generare delle viste materializzate.

Mondrian-JPivot consente di costruire applicazioni di analisi e supporto alle decisioni web-based efficienti e competitive.

Le caratteristiche principali di tale architettura sono:

- Facilità di navigazione
- Visualizzazione grafici sincronizzata con il percorso di navigazione
- Estensibilità interfaccia utente
- Semplicità di integrazione
- Indipendenza dalla piattaforma

Mondrian-JPivot è di sicuro uno strumento più potente di BIRT e JasperReports; rappresenta un vero e proprio strumento analitico che oltre ad offrire funzionalità di reporting implementa importanti risorse di analisi a livello *OLAP*.

Piattaforme Di Business Intelligence

Le piattaforme di Business Intelligence rappresentano dei sistemi molto articolati in cui risiedono diversi moduli software in relazione.

Esistono alcuni progetti open-source tuttora in sviluppo che hanno l'obiettivo di riuscire a realizzare delle organiche e robuste infrastrutture per attività di analisi; tra i progetti più noti risiedono il progetto *SpagoBI* e il progetto *Pentaho*.

Nell'attuale panorama del free software, sono presenti prodotti specifici per aspetti specifici; le piattaforme di Business Intelligence tendono a selezionare i tools più adeguati al fine di realizzare una struttura realmente integrata per la gestione delle informazioni.

Si tratta di progetti ambiziosi e molto articolati, che non nascono quindi come prodotti completi messi a disposizione della comunità, ma piuttosto come idee da sviluppare insieme ad una comunità di interesse. Tali piattaforme integrano quindi strumenti di Business Intelligence di vario genere. Al loro interno risiedono sia strumenti di reportistica che motori OLAP: gli stessi applicativi *JasperReports*, *BIRT* e *Mondrian-JPivot* possono essere inglobati in questi sistemi.

TABELLA COMPARATIVA PRODOTTI BUSINESS INTELLIGENCE OPEN-SOURCE

	PRODOTTI						
CARATTERISTICHE	**JasperReports**	**BIRT**	**Object Visualizer**	**Mondrian-JPivot**	**Greenplum**	**SpagoBI**	**Pentaho**
Funzionalità	Reportistica			Reportistica, OLAP		Reportistica, OLAP, Data Mining	
Flessibilità	Ampia	Buona	Limitata	Ampia	Limitata	Vasta	
Interfaccia Grafica	Possibilità integrazione differenti interfacce	Ben strutturata, interazione intuiviva	Essenziale	Non molto intuitiva	Possibilità di integrazione	Elevate possibilità di integrazione	
Formati aggiuntivi di esportazione risultati	PDF, XLS	PDF	PDF, XLS		PDF	PDF, XLS	
Generazione Grafici	Elevate possibilità		Buone possibilità			Elevate possibilità	

Requisiti	Java, XML, JDBC	Eclipse, Java, AXIS, GEF, EMF, JDBC	Java	Java, Apache Tomcat, JDBC/JConnector	Linux, MAC, Solaris, PostgreSQL	Java, Apache Tomcat, eXo Platform, CMS, JDBC/JConnector	Java, Apache Tomcat, JDBC/Jconnector
Vantaggi principali	Potenzialità di reportistica molto elevate	Ottimo report designer	Procedure wizard per generare query e grafici. Facilità installazione.	Alte capacità di analisi. Facilità di navigazione.	Facilità di installazione. Connettività apera agli strumenti di BI.	Scheduling processi. Vaste possibilità di analisi. Numerose funzionalità.	
Svantaggi principali	Funzionalità OLAP non supportate. Configurazione impegnativa.		Potenzialità ristrette	Comprensione interfaccia grafica non immediata. Configurazione impegnativa.	Incentrato principalmente su PostgreSQL. Non predisposto per sistemi operativi Windows.	Installazione e configurazione decisamente impegnativa. Necessità di integrazione di differenti moduli.	

Il progetto dell'architettura realizzata richiede esplicitamente l'adozione di un motore capace di svolgere analisi OLAP sul data warehouse pianificato. Tale esigenza ha orientato la scelta verso l'architettura *Mondrian-JPivot*; anche se il sistema richiede una fase di configurazione impegnativa, con competenze tecniche elevate, l'applicativo offre altissime potenzialità. L'installazione può essere effettuata senza troppi problemi, il motore OLAP può interagire con qualsiasi tipo di Data Source e l'interfaccia grafica, anche se non immediata, una volta assimilata può essere gestita senza troppe difficoltà.

Gli altri prodotti esaminati da un lato hanno presentato capacità analitiche poco soddisfacenti e strettamente legate alla reportistica, dall'altro hanno visto entrare in gioco piattaforme di analisi altamente sofisticate e spesso difficilmente gestibili; si ritiene quindi che l'architettura Mondrian-JPivot rappresenti il giusto compromesso tra le esigenze di analisi richieste e le funzionalità offerte dall'applicativo.

4.5.4 ARCHITETTURA MONDRIAN-JPIVOT

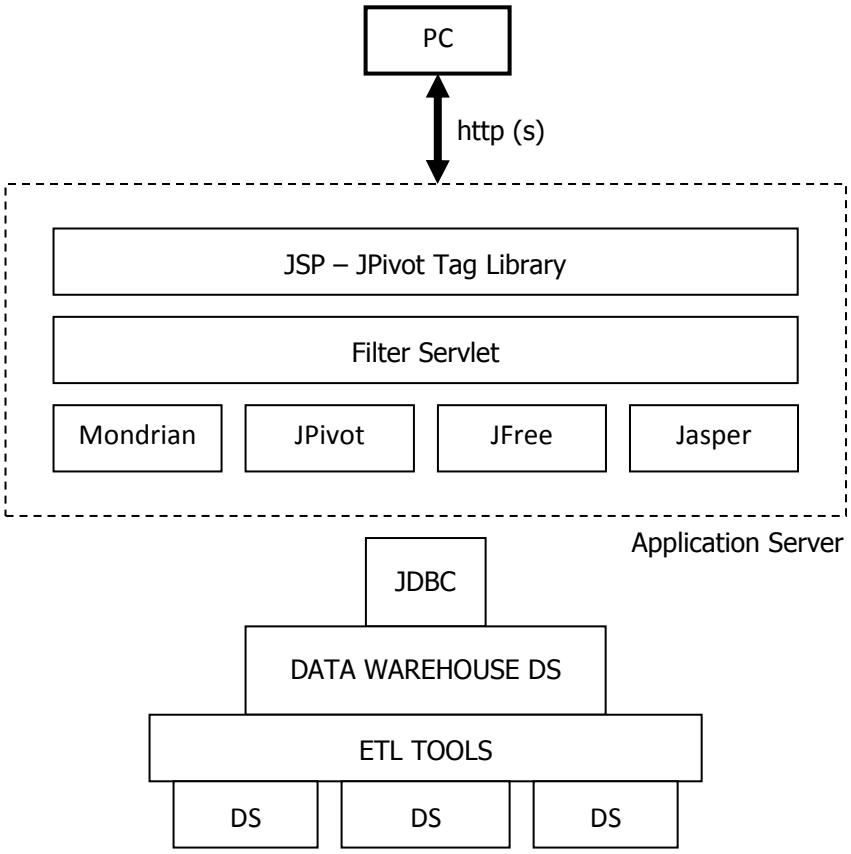
Mondrian è un motore OLAP che consente di gestire modelli multidimensionali interfacciandosi a database relazionali; la connessione alla base di dati di Data Warehouse avviene via JDBC, il che rende indipendente Mondrian dal particolare RDBMS utilizzato. Lo schema multidimensionale della base dati può essere sia star che snowflake, e la sua descrizione viene fornita al motore sotto forma di file XML[28].

Mondrian è progettato per delegare all'RDBMS tutte le funzionalità che questo è in grado di eseguire al meglio, in particolare l'aggregazione e l'utilizzo, ove consentito, di materialized views per ottimizzare la velocità di risposta. Le raffinate strategie di caching consentono buone prestazioni in termini di velocità di esecuzione.

JPivot consente all'utente la navigazione all'interno di una tabella OLAP elaborando dinamicamente query MDX. Utilizza Mondrian come motore OLAP, ma può interagire anche con sorgenti dati XMLA (XML for Analisys). JPivot si basa sulla libreria WCF (Web Component Framework) per il rendering degli oggetti grafici dell'interfaccia utente e sul noto pacchetto JFreeChart per il tracciamento di grafici.

Mondrian-JPivot sono implementati tramite un pool di servlet Java e tag libraries JSP. La comunicazione con l'RDBMS di Data Warehouse avviene via JDBC, il che li rende indipendenti dal database utilizzato e dalla sua collocazione fisica. E' possibile autenticare gli utenti e fornire loro viste parziali e funzionalità di analisi limitate in base ai loro

privilegi. Il deployment avviene in un application server (es: Apache Tomcat) ed è semplice l'integrazione in una web-application preesistente o in un portale.



Caratteristiche dell'architettura:

- Elevata scalabilità: le prestazioni dipendono dall'RDBMS e dall'Application Server adottati
- Flessibilità di installazione: RDBMS, motore OLAP e strato di presentazione possono anche essere installati su macchina diverse
- Indipendenza dal sistema operativo
- Indipendenza dal client (web browser) utilizzato

La configurazione dell'architettura si sviluppa principalmente nei seguenti passi:

- Progettazione e creazione del database
- Editing del file XML per la definizione del data warehouse schema
- Definizione delle query predefinite (istruzioni MDX)
- Testing della connessione al database ed esecuzione delle query

MONDRIAN-JPIVOT – SCHEDA TECNICA DEL SOFTWARE	
PRODUTTORE	Pentaho
LICENZA	Open Source
SITO WEB	http://mondrian.pentaho.org/

REQUISITI

- Java Development Kit & Runtime Environment
- Application Server (es: Apache TOMCAT)
- JDBC/JConnector Driver

FUNZIONALITA'

- Esecuzione query espresse in MDX
- Possibilità di sfruttare Aggregation Table
- Interfaccia grafica per la navigazione nella tabella OLAP

  (slice/dice, drill, roll-up)
- Interazione con sorgenti dati XMLA (XML for Analisys)
- Possibilità di generare grafici
- Possibilità di esportare risultati di una query in formato PDF o XLS

PRO

- Vantaggio economico (licenza open source)
- Alte capacità di analisi
- Facilità di navigazione
- Flessibilità
- Comunicazione con sistemi esterni (es: Excel)
- Indipendenza dal tipo di database utilizzato

CONTRO

- Configurazione piuttosto impegnativa per ogni problema di studio
- Richiede competenze tecniche per la personalizzazione delle query
- Comprensione interfaccia utente non immediate

LINK UTILI

- http://sourceforge.net/projects/mondrian/
  (pagina dedicata allo sviluppo del progetto Mondrian)
- http://jpivot.sourceforge.net/
  (pagina dedicata alla sviluppo del profetto JPivot)
- http://www.miriade.it
  (società di consulenza informatica: possibilità di accedere
  ad articoli e recensioni su prodotti Business Intelligence Open Source)
- http://www.compiere.org
  (progetto parallelo open-source ERP & CRM solution)

4.6 CONFIGURAZIONE DEL SERVER OLAP

Come già esposto, il software scelto per implementare le funzionalità analitiche previste per l'architettura è il *Mondrian-JPivot*.

In fase di realizzazione si è scesi nel dettaglio per comprendere come funziona lo strumento, in che modo interagisce con il database in esame e come deve essere configurato per adattare le sue potenzialità alle esigenze dell'utente.

Il server OLAP Mondrian è un progetto in continuo sviluppo, inizialmente viene presentato con una configurazione standard predisposta per interagire con un demo database realizzato in Microsoft Access, in questo modo l'utente può testare l'applicativo nel suo pieno funzionamento stimando tutte le potenzialità che offre.

La parte più difficile da affrontare riguarda la configurazione di Mondrian per operare su altri database, l'adattamento dell'applicativo alla sorgente dati da esaminare deve tener conto di ogni aspetto: dal tipo di database, alla specifica struttura delle tabelle che lo compongono.

Il Mondrian lavora sulla base di una continua interazione di servlet Java (utilizzo di Java Server Page). Il prodotto si presenta come parte integrante di una web-application: l'applicazione per poter funzionare necessita di un application server in cui essere pubblicata ed attivata consentendo così l'accesso al servizio con interazione tramite browser.

Il pacchetto di istallazione include tutte le risorse di cui si necessita:

- La libreria per l'integrazione delle funzionalità di Mondrian con applicazioni Java (mondrian.jar).
- La documentazione completa per l'installazione, la configurazione e il design personalizzato di nuove applicazioni.
- Le risorse per attivare una demo del prodotto e testare le sue funzionalità (database Access, file XML, script SQL).
- Un Web-Archive file (mondrian.war) da utilizzare per effettuare il deploy dell'applicazione nel web server Tomcat.

Una volta effettuato il deploy nel web server Tomcat è possibile accedere all'applicativo servendosi di un web browser; il pacchetto di installazione Mondrian implementa già al suo interno la possibilità di utilizzare l'interfaccia JPivot per navigare all'interno delle tabelle OLAP.

La parte più impegnativa si presenta nel momento in cui si ha la necessità di adattare Mondrian ad un proprio caso di studio; ogni file di configurazione presente nel pacchetto è settato per supportare l'interazione con il database demo: sia le stringhe di connessione, sia le query di default sono scritte riferendosi alla base dati di prova.

E' fondamentale sapere che Mondrian utilizza un file XML per descrivere la struttura del data warehouse su cui operare: attraverso questo file è possibile identificare **Cubi**, **Dimensioni**, **Misure**, **Gerarchie**, **Livelli** etc. Ciò permette di esprimere dei riferimenti da utilizzare nella scrittura delle istruzioni MDX e consente a JPivot di poter realizzare la navigazione all'interno della tabella OLAP (in pratica lo stato della tabella OLAP non è altro che la rappresentazione grafica del risultato di una query MDX); il modello XML esprime lo schema del database multi-dimensionale.

Il pacchetto Mondrian-JPivot prevede per il demo database un file XML predefinito che esprime in maniera esaustiva le caratteristiche del data warehouse.

Il file è piuttosto elaborato e definisce le varie dimensioni e i vari cubi multi-dimensionali associati al database, tutto questo usufruendo di determinati tag che consentono la modellazione e la tipizzazione dei campi e delle tabelle della base di dati. Gli attributi associati ai tag consentono inoltre di stabilire le caratteristiche associate agli oggetti logici che realizzano lo schema ed esprimere le misure integrate nei cubi.

4.6.1 CONNESSIONE AL DATABASE

Nel momento in cui ci si trova ad operare con un nuovo database occorre seguire diversi stadi di configurazione del software per impostare la comunicazione con tra il motore OLAP ed il database fisico; a tal proposito si deve intervenire come segue:

- **Modificare** ogni **stringa di connessione** in modo da puntare il riferimento al database di interesse (la stringa di connessione è strettamente correlata al tipo di database utilizzato).
- **Editare un nuovo file XML** per descrivere il data warehouse schema.
- **Modificare le query** predefinite in modo coerente, sulla base degli oggetti logici definiti precedentemente nel file XML.

L'editing del file XML è una delle operazioni più delicate da affrontare. Il file deve essere scritto in maniera coerente e sintatticamente corretta.

Naturalmente l'elaborazione del file deve essere un'operazione graduale: inizialmente si definiscono semplici relazioni, successivamente si arricchisce la struttura dello schema in modo da raggiungere livelli di complessità sempre più elevati.

Nel caso di studio in esame, Mondrian deve interfacciarsi con un database appositamente progettato per supportare il monitoraggio e l'analisi di prezzi e tariffe. La base dati è gestita attraverso *mySQL*; questo implica che ogni stringa di connessione definita in Mondrian debba essere configurata in modo da puntare al database di riferimento. In primis, la preoccupazione principale è testare un corretto dialogo tra database mySQL e Mondrian in modo da porre le basi per la costruzione di successive configurazioni più complesse.

Il processo di implementazione si articola nei seguenti passi:

1. Configurazione di Mondrian per l'interazione con mySQL Server
2. Design del Mondrian schema (editing del file XML relativo al db di interesse)
3. Definizione di una query MDX di default
4. Verifica del funzionamento della connessione al db e della corretta esecuzione della query predefinita
5. Sviluppo e collaudo di elaborazioni successive

Interfaccia Verso il Database MySQL

Mondrian richiede che alcuni file vengano configurati in modo appropriato per consentire un'interazione corretta con il database mySQL di interesse.

E' possibile gestire il dialogo con il database attraverso la libreria Java *mysql-connector* che si occupa di gestire gli aspetti di connessione ed esecuzione delle interrogazioni.

Le modifiche da apportare a Mondrian per far si che comunichi correttamente con il database mySQL sono:

- Modifica del file ***mondrian.properties***
- Modifica del file ***datasources.xml***
- Modifica del file ***web.xml***
- Inclusione della libreria ***mysql-connector-java*** nella directory delle librerie

Tutti questi file sono individuati partendo dalla cartella di riferimento *$TOMCAT_HOME/webapps/mondrian/WEB-INF*.

Una volta editati correttamente si può procedere con la definizione del file XML che esprime la rappresentazione dello schema del data warehouse, infine si definiranno le query predefinite da processare.

Le query sono riportate all'interno di file JSP presenti nella directory *queries*, nella stessa directory va anche allocato il file XML creato (esempi di file jsp da modificare sono: ***mondrian.jsp, arrows.jsp, colors.jsp, fourhier.jsp***).

Settaggio Dei Files

Nel file **mondrian.properties** vengono impostate le principali informazioni di configurazione di Mondrian, permette cioè di specificare i parametri cui far riferimento in fase di esecuzione. Possono essere espresse numerose proprietà sulla base delle quali verranno gestite tutte le operazioni del sistema.

Una di queste proprietà permette di specificare la stringa di connessione da caricare, è opportuno quindi inserire o modificare all'interno del file il settaggio idoneo.

Alla base dati in cui verranno registrate le voci di prezzo è stato associato il nome *warehouse_prezzi*; ciò significa che, all'interno del file mondrian.properties dovrà essere esplicitato un settaggio di questo genere:

```
mondrian.test.connectString= Provider=mondrian;
JdbcDrivers=com.mysql.jdbc.Driver;
Jdbc=jdbc:mysql://localhost:3306/warehouse_prezzi?user=root;
Catalog=/WEB-INF/queries/Prezzi.xml;
```

Come osservabile, oltre ad esplicitare i riferimenti rispetto al driver ed alla stringa di connessione, nell'impostazione viene anche espresso il riferimento che punta al file *Prezzi.xml*, che rappresenta il file in cui è dichiarato lo schema logico del data warehouse.

Effettuando il parsing del file, l'applicativo è quindi in grado di riconoscere i riferimenti al database e le proprietà associate agli oggetti logici definiti (cubi, dimensioni, livelli etc.).

Lo stesso discorso vale per i file **datasource.xml** e **web.xml**, le stringhe di connessione devono essere scritte in conformità al database di riferimento (quindi a quanto definito nel file *mondrian.properties*).

Nota:

Nella configurazione sopra esposta, si è utilizzata la seguente stringa di connessione:

```
jdbc:mysql://localhost:3306/warehouse_prezzi?user=root
```

Questo perché, il file di configurazione fa riferimento ad un settaggio gestito in locale in cui la configurazione del server per la connessione al database richiede come nome utente "*root*" e non prevede alcuna password di accesso; la porta di riferimento è la 3306. Ma in generale la stringa di connessione assume la seguente struttura:

```
jdbc:mysql://HOSTNAME:PORT/DATABASE;
jdbcUser=USERNAME; jdbcPassword=PASSWORD;
```

dove USERNAME e PASSWORD rappresentano rispettivamente il nome utente e la password da utilizzare per accedere al DATABASE scelto, mentre HOSTNAME e PORT specificano l'indirizzo di connessione.

Libreria **mysql-connector-java:** Per far in modo che Mondrian riconosca il driver jdbc, che consente il dialogo con il database mySQL, è necessario che la libreria *mysql-connector-java.jar* sia inclusa nel classpath; si tratta di un Java Archive file che ingloba le classi necessarie per gestire l'interfacciamento tra Java ed un database mySQL.

.

.

.

4.6.3 SCRITTURA DEL MONDRIAN XML SCHEMA

La fase di scrittura dello schema XML per la descrizione del modello del data warehouse rappresenta un passo cruciale da affrontare per rendere operativa l'architettura *Mondrian-JPivot;* la definizione dello schema deve essere fatta con molta cura in modo da rispettare le esigenze di analisi previste ed in modo da rispettare la conformità con il modello relazionale precedentemente pianificato.

Uno schema definisce un database multidimensionale; contiene il modello logico (costituito da cubi, gerarchie e membri) e le relazioni con il modello fisico.

Lo schema XML è stato sviluppato tenendo conto delle maggiori esigenze di analisi possibili, introducendo anche dimensioni fittizie a supporto delle interrogazioni più elaborate.

Grazie al modello XML è infatti stato possibile risolvere alcune problematiche riscontrate in fase operativa; il corrispondente modello XML derivato è consultabile nelle seguenti pagine.

```
<?xml version="1.0"?>
<Schema name="Prezzi">

<!-- Tempo (Time) -->
<Dimension name="Time" type="TimeDimension">
  <Hierarchy hasAll="false" primaryKey="time_id">
    <Table name="time"/>
    <Level name="Anno" column="anno" type="Numeric" uniqueMembers="true" levelType="TimeYears"/>
    <Level name="Quadrimestre" column="quad" uniqueMembers="false" levelType="TimeQuarters"/>
    <Level name="Mese" column="mese" uniqueMembers="false" levelType="TimeMonths"/>
      <Level name="Settimana" column="settimana" uniqueMembers="false" levelType="TimeWeeks"/>
      <Level name="Giorno" column="giorno" type="Numeric" uniqueMembers="false" levelType="TimeDays"/>
  </Hierarchy>
</Dimension>

<!-- Tipo (Property) -->
<Dimension name="Tipo">
  <Hierarchy hasAll="false" primaryKey="prop_id">
            <Table name="proprieta"/>
            <Level name="Oggetto" column="feature_01" uniqueMembers="true">
                  <Property name="Classe" column="feature_03"/>
                  <Property name="Sottoclasse" column="feature_02"/>
            </Level>
  </Hierarchy>
</Dimension>
```

Mondrian XML Schema

```
<!-- Classe Prodotto -->
<Dimension name="Classe Prodotto">
   <Hierarchy hasAll="true" allMemberName="Tutte Le Classi" primaryKey="prop_id">
     <Table name="proprieta"/>
        <Level name="Tipologia" column="feature_03" uniqueMembers="true"/>
     <Level name="Caratteristica" column="feature_02" uniqueMembers="false"/>
   </Hierarchy>
</Dimension>

<!-- Marchio (Brand) -->
<Dimension name="Marchio">
      <Hierarchy hasAll="true" allMemberName="Tutti I Marchi" primaryKey="marchio_id">
               <Table name="marchio"/>
               <Level name="Nome" column="nome" uniqueMembers="true"/>
   </Hierarchy>
</Dimension>

<!-- Sorgenti dati (Source) -->
<Dimension name="Fonte">
      <Hierarchy hasAll="true" allMemberName="Tutte Le Fonti" primaryKey="fonte_id">
               <Table name="fonte"/>
               <Level name="Sito" column="url" uniqueMembers="true"/>
   </Hierarchy>
</Dimension>
```

Mondrian XML Schema

```
<!-- Dislocazione (Dislocation) -->
<Dimension name="Dislocazione" foreingKey="disloca_id">
      <Hierarchy hasAll="false" primaryKey="disloca_id" primaryKeyTable="dislocazione">
       <Join leftKey="reg_id" rightKey="reg_id">
        <Table name="dislocazione" />
              <Table name="regioni" />
       </Join>
       <Level name="Nazione" table="dislocazione" column="nazione" uniqueMembers="true"/>
       <Level name="Regioni" table="regioni" column="regione" uniqueMembers="true" hideMemberIf="IfBlankName"/>
       <Level name="Province" table="dislocazione" column="nome_prov" uniqueMembers="true" hideMemberIf="IfBlankName"/>
       <Level name="Comuni" table="dislocazione" column="nome_comune" uniqueMembers="false" hideMemberIf="IfBlankName">
              <Property name="Agg_Naz" column="livello"/>
       </Level>
  </Hierarchy>
</Dimension>

<!-- Livello Aggregazione Dislocazione (Necessario per filtrare il prezzo medio aggregato) -->
<Dimension name="Livello Aggregazione">
  <Hierarchy hasAll="true" allMemberName="Tutti I Livelli" primaryKey="disloca_id">
    <Table name="dislocazione"/>
      <Level name="Gruppo" column="livello" uniqueMembers="true"/>
  </Hierarchy>
</Dimension>
```

Mondrian XML Schema

```
<!-- Definizione Cubo -->

<Cube name="Report">
  <Table name="prezzi"/>
  <DimensionUsage name="Time" source="Time" foreignKey="time_id"/>
  <DimensionUsage name="Tipo" source="Tipo" foreignKey="prop_id"/>
  <DimensionUsage name="Marchio" source="Marchio" foreignKey="marchio_id"/>
  <DimensionUsage name="Dislocazione" source="Dislocazione" foreignKey="disloca_id"/>
  <DimensionUsage name="Fonte" source="Fonte" foreignKey="fonte_id"/>
  <DimensionUsage name="Livello Aggregazione" source="Livello Aggregazione" foreignKey="disloca_id"/>

  <!-- Misure senza aggregatore -->
  <Measure name="Prezzo Medio (Nessuna Aggregazione)" column="prezzo_med" aggregator="none" formatString="#,##0.000" visible="false"/>
  <Measure name="Prezzo Max (Nessuna Aggregazione)" column="prezzo_max" aggregator="none" formatString="#,##0.000" visible="false"/>
  <Measure name="Prezzo Min (Nessuna Aggregazione)" column="prezzo_min" aggregator="none" formatString="#,##0.000" visible="false"/>

  <!-- Misure con aggregazione (Avg) -->
  <Measure name="Prezzo Medio (Aggregato)" column="prezzo_med" aggregator="avg" formatString="#,##0.000" visible="true">
      <Property name="Data Caricamento" column="data_caricamento"/>
  </Measure>

  <!-- Misure con aggregazione (Max) -->
  <Measure name="Prezzo Massimo" column="prezzo_max" aggregator="max" formatString="#,##0.000" visible="true">
      <Property name="Data Caricamento" column="data_caricamento"/>
  </Measure>
```

Mondrian XML Schema

```
 <!-- Misure con aggregazione (Min) -->
<Measure name="Prezzo Minimo" column="prezzo_min" aggregator="min" formatString="#,##0.000" visible="true">
    <Property name="Data Caricamento" column="data_caricamento"/>
</Measure>

<!-- Prezzo Medio: Filtraggio prezzi in riferimento al livello di aggregazione -->
<CalculatedMember name="Media Rilevamenti Nazionali" dimension="Measures"
formula="COALESCEEMPTY((Measures.[Prezzo Medio (Aggregato)], [Livello Aggregazione].[Nazionale]), NULL)" visible="false"/>
<CalculatedMember name="Media Rilevamenti Regionali" dimension="Measures"
    formula="COALESCEEMPTY((Measures.[Prezzo Medio (Aggregato)], [Livello Aggregazione].[Regionale]), NULL)" visible="false"/>
<CalculatedMember name="Media Rilevamenti Provinciali" dimension="Measures"
    formula="COALESCEEMPTY((Measures.[Prezzo Medio (Aggregato)], [Livello Aggregazione].[Provinciale]), NULL)" visible="false"/>
<CalculatedMember name="Media Rilevamenti Comunali" dimension="Measures"
    formula="COALESCEEMPTY((Measures.[Prezzo Medio (Aggregato)], [Livello Aggregazione].[Comunale]), NULL)" visible="false"/>

<!-- Query MDX Caricamento Prezzi Medi in riferimento al livello Dislocazione -->
<CalculatedMember name="Prezzo Medio (Estratto)" dimension="Measures" visible="true">
    <Formula>
        iif([Dislocazione].CurrentMember.Level IS [Dislocazione].[Nazione],[Measures].[Media Rilevamenti Nazionali],
        iif([Dislocazione].CurrentMember.Level IS [Dislocazione].[Regioni],[Measures].[Media Rilevamenti Regionali],
        iif([Dislocazione].CurrentMember.Level IS [Dislocazione].[Province],[Measures].[Media Rilevamenti Provinciali],
        iif([Dislocazione].CurrentMember.Level IS [Dislocazione].[Comuni],[Measures].[Media Rilevamenti Comunali],NULL))))
    </Formula>
</CalculatedMember>
</Cube></Schema>
```

Mondrian XML Schema

La fase di configurazione del *Mondrian XML Schema* ha messo in evidenza come il linguaggio predisposto dall'applicativo sia altamente adattabile alle diverse esigenze di analisi. Nel sito di riferimento del progetto è disponibile la documentazione necessaria per comprendere il significato ed il corretto utilizzo dei tag che definiscono lo schema.

Come osservabile dallo schema riportato, il modello XML si basa sui seguenti tag:

- **Cube** = collezione di misure e dimensioni
- **Table** = definisce le relazioni con le tabelle del modello fisico
- **Dimension** = specifica ogni dimensione di riferimento, le dimensioni possono essere associate ad un unico cubo oppure condivise (utilizzate da più cubi)
- **Hierarchy** = definisce le gerarchie legate ad una dimensione
- **Level** = definisce un livello per una data gerarchia
- **Measures** = esprime le misure di interesse; ogni misura ha un nome che identifica la colonna nella fact table ed un aggregatore.

  L'aggregatore può essere:

  - *Sum* (somma)
  - *Avg* (media)
  - *Max* (massimo)
  - *Min* (minimo)
  - *Count* (conteggio)
  - *Distinct Count* (conteggio distinto)

La strategia di aggregazione usata da Mondrian è la seguente[32]:

- La fact table è memorizzata nel DBMS
- Per aggregare i dati viene utilizzata la clausola *group by* dell'SQL
- Se l'RDMBS supporta viste materializzate e l'amministratore del database sceglie di creare viste materializzate per particolari aggregazioni, allora *Mondrian* usa esse per semplicità.

Partendo dallo schema multidimensionale possiamo sviluppare le seguenti definizioni[32]:

- Definiamo membro un punto in cui le dimensioni sono determinate da un preciso insieme di valori degli attributi
- Una gerarchia è un insieme di membri organizzati in una struttura per convenienza di analisi.
- Un livello è una collezione di membri che hanno la stessa distanza dalla radice della gerarchia.
- Una dimensione è una collezione di gerarchie che vengono discriminate dallo stesso attributo della fact table.

Come è possibile osservare, per ragioni di uniformità, le *misure* sono trattate come un membro di una speciale dimensione chiamata "Measures"; l'attributo opzionale *formatString* specifica come il valore dell'attributo deve essere visualizzato.

Dall'analisi dello schema *XML* riportato è possibile individuare della misure fittizie in cui viene utilizzato l'aggregatore "*none"*, questo perché a livello implementativo è stata utilizzata la versione ricompilata di *Mondrian* che, oltre a gestire i fattori di aggregazione predefiniti, integra anche l'aggregazione nulla. L'aggregatore "*none"* comunque non ricopre alcun ruolo di rilievo a livello operativo.

Osservando il modello XML è anche possibile riscontrare un massiccio uso dei membri calcolati. Il tag **CalculatedMember** permette di processare delle query ad hoc per derivare delle misure o dei livelli calcolati secondo le necessità di sviluppo. L'uso connesso dei *CalculatedMember* con le istruzioni *MDX* consente di risolvere eventuali problematiche legate alla gestione dei dati; ciò naturalmente implica che, chi si occupa della definizione dello schema, abbia le competenze necessarie per scrivere un modello valido e sia in grado di elaborare eventuali query *MDX* a supporto dell'analisi.

Proprio attraverso i *CalculatedMember* è stato possibile risolvere la problematica riguardate la gestione dei dati pre-aggregati.

Il modello XML definito prevede due misure di riferimento per il prezzo medio:

- **Prezzo Medio (Aggregato)**
- **Prezzo Medio (Estratto)**

Il "*Prezzo Medio (Aggregato)*" non rappresenta altro che il valore derivato dalla media di tutti i prezzi medi registrati. Il riferimento a tale misura però deve essere gestito con cautela; infatti, per avere un valore significativo del "*Prezzo Medio (Aggregato)*" è opportuno che sia specificato un "*Livello di Aggregazione*" di riferimento.

Questo perché, considerando tutti i livelli di aggregazione, il "*Prezzo Medio (Aggregato)*" risulterebbe come la media di tutti le voci di prezzo registrate ai vari livelli; ad esempio, in alcuni casi, potrebbe essere definito come la media di un valore letto a livello nazione con un valore letto a livello comunale. Una situazione di questo genere produrrebbe una misura poco attendibile e logicamente non corretta; per tale motivo è necessario che nel momento in cui ci si trovi ad operare con il "*Prezzo Medio (Aggregato)*" sia necessariamente esplicitato un "Livello di Aggregazione" cui far riferimento.

Il "*Prezzo Medio (Aggregato)*" resta comunque una misura importate da poter sfruttare perché consente di sviluppare degli interessanti confronti. Se, ad esempio, si prende come riferimento di analisi il livello "*Nazionale*", attraverso il "*Prezzo Medio (Aggregato)*" è possibile confrontare i valori ottenuti dalla media di tutti i rilevamenti comunali (a disposizione per l'intera nazione) con eventuali valori ottenuti dalla media di rilevamenti per diversi livelli di intervento (regionale o provinciale).

Il "*Prezzo Medio (Estratto)"* è stato definito proprio per riuscire ad avere a disposizione una misura in grado di visualizzare ad ogni livello della dimensione *DISLOCAZIONE* il relativo valore pre-aggregato estratto dalla sorgente web.

Questo significa che, se la fonte mette a disposizione dei rilevamenti comunali, provinciali, regionali e nazionali, nel momento in cui si naviga la tabella *OLAP* il livello corrente della *DISLOCAZIONE* deve visualizzare direttamente il dato disponibile per quel dato livello, senza ricavarlo da aggregazioni calcolate. Un'esigenza di questo genere può essere gestita sfruttando delle operazioni di filtraggio e dei controlli condizionali applicabili attraverso le query *MDX*.

In tal senso è possibile vedere come nello schema XML la misura "*Prezzo Medio (Estratto)*" venga definita nel seguante modo:

```
<CalculatedMember name="Prezzo Medio (Estratto)" dimension="Measures" visible="true">
  <Formula>
      iif([Dislocazione].CurrentMember.Level IS [Dislocazione].[Nazione],
      [Measures].[Media Rilevamenti Nazionali],
      iif([Dislocazione].CurrentMember.Level IS [Dislocazione].[Regioni],
      [Measures].[Media Rilevamenti Regionali],
      iif([Dislocazione].CurrentMember.Level IS [Dislocazione].[Province],
      [Measures].[Media Rilevamenti Provinciali],
      iif([Dislocazione].CurrentMember.Level IS [Dislocazione].[Comuni],
      [Measures].[Media Rilevamenti Comunali],NULL))))
  </Formula>
</CalculatedMember>
```

Tradotto in parole "povere" un'istruzione di questo genere comunica al sistema di visualizzare il valore che si riferisce al corrente livello di dislocazione: "se ti trovi ad un livello comunale visualizza la media dei rilevamenti comunali", "se ti trovi ad un livello provinciale visualizza la media dei rilevamenti provinciali" e così via. Le medie dei rilevamenti comunali, provinciali, regionali e nazionali sono calcolate a priori, attraverso delle operazioni di filtraggio applicate sulla base del *"Livello di Aggregazione"*.

E' da evidenziare il fatto che la possibilità di filtrare il prezzo medio rispetto al livello di aggregazione è data dall'inserimento di una informazione aggiuntiva (campo *livello*) nella tabella delle dislocazioni; tale informazione esplicita chiaramente il livello (nazionale, regionale, provinciale o comunale) che caratterizza ogni singolo identificativo.

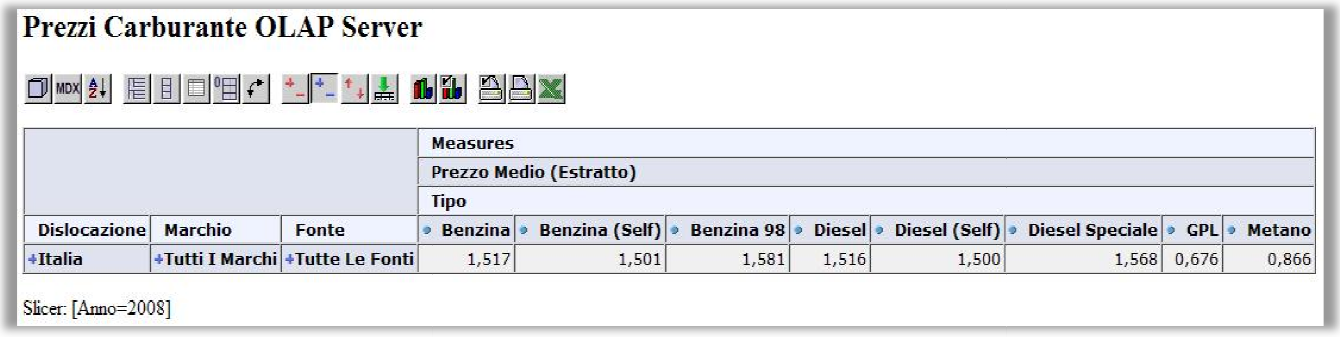
Un intervento a livello logico come quello appena esposto risulta sicuramente di non immediata comprensione ma rappresenta indubbiamente la soluzione più elegante per riuscire a gestire dati pre-aggregati a livello di *DISLOCAZIONE*; si riesce infatti con questa tecnica a discriminare la visualizzazione del prezzo medio per i diversi livelli di dislocazione senza intaccare le potenzialità di aggregazione per le altre dimensioni in uso.

Dopo aver definito lo schema XML l'ultimo passo da affrontare consiste nell'impostare una query di default da utilizzare come istruzione MDX di base per inizializzare la tabella OLAP visualizzata tramite *JPivot*. Questa query è definita nel file **mondrian.jsp** presente nella directory *queries*. In riferimento al data warehouse implementato, il file *mondrian.jsp* può essere impostato nel seguente modo:

```
<jp:mondrianQuery id="query01" jdbcDriver="com.mysql.jdbc.Driver"
jdbcUrl="jdbc:mysql://localhost:3306/warehouse_prezzi?user=root"
catalogUri="/WEB-INF/queries/Prezzi.xml">
select Crossjoin({[Measures].[Prezzo Medio (Estratto)]}, {[Tipo].Members}) ON COLUMNS,
 {([Dislocazione].[Italia], [Marchio].[Tutti I Marchi], [Fonte].[Tutte Le Fonti])} ON ROWS
from [Report]
where [Time].[2008]
</jp:mondrianQuery>
<c:set var="title01" scope="session">Prezzi Carburante OLAP Server</c:set>
```

A questo punto si può avviare *JPivot* e testare il corretto funzionamento dell'applicazione eseguendo operazioni di navigazione sulla tabella (ogni operazione di navigazione corrisponde ad una determinata query MDX).

Avviando la *JPivot Table* sarà richiamata l'esecuzione della query definita nel file *mondrian.jsp* e il risultato dell'elaborazione si concretizzerà nella seguente visualizzazione:



Prezzi Carburante OLAP Server

			Measures							
			Prezzo Medio (Estratto)							
			Tipo							
Dislocazione	Marchio	Fonte	Benzina	Benzina (Self)	Benzina 98	Diesel	Diesel (Self)	Diesel Speciale	GPL	Metano
+Italia	+Tutti I Marchi	+Tutte Le Fonti	1,517	1,501	1,581	1,516	1,500	1,568	0,676	0,866

Slicer: [Anno=2008]

Partendo da questa visualizzazione iniziale si può procedere diramando e modificando l'albero di navigazione in modo da suddividere, aggregare e visualizzare i dati secondo le proprie esigenze.

Anche i file *arrows.jsp***,** *colors.jsp***,** *fourhier.jsp*, presenti nella stessa directory, possono essere modificati per il nuovo database in uso, ma ciò è necessario solo se si richiedono operazioni più sofisticate e non è indispensabile ai fini del funzionamento di *JPivot*. Nella directory *$TOMCAT_HOME/webapps/mondrian/*, invece, sono definiti alcuni file da modificare per avere un'interfaccia di analisi più funzionale.

I file in questione sono: *adhoc.jsp***,** *pivot.jsp***,** *taglib.jsp***.** In particolare il file adhoc.jsp consente di predisporre una serie di query predefinite da poter richiamare e processare a proprio piacimento senza dovere ogni volta ri-editare l'interrogazione MDX di cui si necessita.

Tutte queste personalizzazioni consento di modellare il sistema di analisi adattandolo alle proprie esigenze così da avere uno strumento di calcolo flessibile ed efficiente.

CONCLUSIONI

La tesi presentata ha descritto quelle che sono state le fasi di progettazione e sviluppo di un'architettura orientata alle analisi sui prezzi. Si è dimostrato come sia possibile automatizzare un processo per il riuso delle informazioni attualmente disponibili sul web e come si possano integrare i flussi di dati in maniera omogenea verso un modello comune. Il progetto si è poi occupato di riversare il patrimonio informativo raccolto verso tecnologie di data warehousing che potessero costituire una solida base per l'elaborazione di analisi avanzate.

I requisiti che hanno guidato l'intero ciclo di sviluppo miravano alla costruzione di una soluzione flessibile ed efficiente, che potesse essere adattata a nuove situazione di analisi senza ricorre ad interventi a basso livello. Tale presupposto ha condotto verso la definizione di un linguaggio di riferimento a supporto di un algoritmo interpretativo per trattare i dati estratti dalle diverse sorgenti web.

L'architettura realizzata consente di configurare delle routine di estrazione per più pagine web che riproducono differenti report sui prezzi; il modulo *ETL* sviluppato sfrutta un modello di metadati attraverso il quale è possibile associare ad ogni singola informazione estratta il giusto significato, in modo che possa essere caricata coerentemente all'interno del *data warehouse* implementato. L'*ETL* si occupa inoltre della supervisione della fase di caricamento dei dati; applicando una serie di controlli il software riesce a rilevare eventuali errori ed incoerenze tra le informazioni trattate, segnalando i problemi riscontrati in specifici file di log.

La fase di sperimentazione ha evidenziato come il sistema riesca a coordinare delle estrazioni, avviate periodicamente, in modo efficiente e valido; durante il periodo di test l'architettura non ha presentato alcuna problematica di rilievo ed il caricamento dei dati è stato eseguito senza che si siano appurate complicazioni nell'applicazione dell'algoritmo interpretativo.

E' chiaro che l'attività di configurazione debba essere affidata ad un amministratore che abbia le dovute competenze per interagire con il linguaggio di sviluppo e sia capace di comprendere ed applicare la logica interpretativa; chi si occupa della configurazione del sistema deve quindi essere abile ad analizzare la natura delle pagine web e di conseguenza configurare delle routine di estrazione e caricamento efficaci.

L'intera attività di configurazione ed alimentazione del data warehouse rimane comunque trasparente all'utilizzatore finale (analista) che accede al servizio tramite web browser e può usufruire degli strumenti di analisi per navigare la tabella *OLAP* ed applicare tutte le possibili combinazioni di filtraggio sui dati.

La fase di testing ha inoltre messo in evidenza come, anche prendendo in considerazione una sola fonte, si riescano a strutturare le informazioni estratte in modo da raggiungere un'elasticità analitica che supera di gran lunga le potenzialità offerte dal servizio stesso da cui i dati sono stati letti; spesso i siti riportano le informazioni in maniera poco strutturata senza includere una gestione efficiente dello storico o prevedere comparazioni personalizzate tra i dati, l'utilizzo di un data warehouse permette di superare tali limiti.

Anche considerando poche fonti di riferimento, in fase di sperimentazione si è visto come, ad ogni estrazione, sia possibile alimentare il data warehouse con migliaia di record informativi generati automaticamente dai dati prelevati.

E' da sottolineare come il data warehouse sia stato progettato in maniera lineare e facilmente estendibile, il fatto che si sia ricorso al recupero delle informazioni dalle sorgenti web non esclude che la banca dati possa essere alimentata in altre modalità: inserimento diretto, interpretazione fogli elettronici, interpretazione documenti pdf, etc. La fase di sperimentazione si è concentrata sui prezzi dei carburanti, ma si è anche visto come sia possibile indirizzare l'architettura verso voci di prezzo relative ad altre tipologie di prodotti (Alimenti, Servizi, etc.).

L'intera attività progettuale ha inoltre fornito una serie di spunti per possibili sviluppi futuri. L'analisi dello stato attuale del progetto suggerisce diversi aspetti che meriterebbero un approfondimento; a partire da una possibile revisione del codice fino a giungere all'implementazione di tool grafici che possano migliorare l'usabilità del sistema.

A livello grafico si potrebbe intervenire sviluppando delle interfacce efficacemente strutturate per agevolare sia la gestione delle analisi, sia le fasi di configurazione. Da questo punto di vista si potrebbe prevedere la progettazione di un'interfaccia web opportunamente modellata per agevolare i percorsi analitici, attraverso la definizione di query predefinite o la selezione di specifici ambiti di controllo; a livello di configurazione si potrebbe invece prevedere la generazione di una serie di utility per supportare le operazioni di settaggio delle routine *ETL*.

Un ulteriore spunto per futuri interventi riguarda l'idea di studiare un metodo per gestire gli aggiornamenti della tabella in cui sono registrate le *dislocazioni*. Attualmente il web mette a disposizione molti servizi indirizzati verso informazioni geografiche (es. *Google Maps*, *Live Search*, etc.), sarebbe interessante predisporre un modulo che consenta di interfacciare la tabella delle dislocazioni con tali servizi in modo da coordinare eventuali aggiornamenti o integrare informazioni più dettagliate.

Un altro aspetto, cui sarebbe necessario dedicare un approfondimento, riguarda l'eventuale possibilità di sviluppare un componente software esterno che abbia il compito di esaminare periodicamente le informazioni registrate nel data warehouse per cercare di valutare la veridicità dei dati inseriti. L'applicativo potrebbe basarsi su misure statistiche per segnalare eventuali dati non fedeli alla classe informativa cui appartengono (es. voci di prezzo con ordine di grandezza non conforme agli standard); in sostanza si svolgerebbe una verifica, a posteriori della fase di caricamento, per consolidare i dati inseriti.

Parallelamente a tutto ciò è auspicabile che l'architettura, per migliorare dal punto di vista dell'usabilità e delle performance, sia sottoposta ad aggiornamenti e siano previste delle attività interamente concentrate al miglioramento del codice per il modulo ETL. Analogamente, per quanto riguarda l'ambito analitico, sarebbe necessario avviare una lunga e ben documentata attività di ricerca allo scopo di inquadrare le tipologie di analisi più significative ed eventualmente definire potenziali misure calcolate da integrare a livello logico nello schema del data warehouse (es. variazioni percentuali, indici di significatività).

Questi sono solamente alcuni dei possibili interventi applicabili per migliorare l'architettura realizzata; attualmente il sistema offre una solida base ben funzionante che implementa una logica originale per interpretare dati estratti dalle pagine web. Tale prospettiva apre le porte ad una serie di possibili future implementazioni con lo scopo di ampliare il sistema per riuscire a costituire una completa ed efficiente piattaforma di business intelligence.

BIBLIOGRAFIA

[1] Provincia Di Bolzano - *www.provinz.bz.it/osservatorio-prezzi*, LAST VIEW: Ottobre 2007

[2] Ministero Dello Sviluppo Economico - *www.osservaprezzi.it*, LAST VIEW: Ottobre 2007

[3] Provincia Di Rovigo - *Osservatorio Provinciale dei Prezzi – Bollettino Statistico degli Indici dei Prezzi al Consumo*, Aprile 2004

[4] A. Pace - *"Analisi e Ingegnerizzazione di un Processo di Estrazione Dati da Web",* Seminario Sistemi Informativi, Università Degli Studi Roma Tre, 2006

[5] R. Pressman - *"Principi di Ingegneria del Software"* 4°ed., McGraw-Hill, 2004

[6] P. Atzeni, S. Ceri, P. Fraternali, S. Paraboschi, R. Torlone - *"Basi di Dati – Architetture e Linee di Evoluzione"*, McGraw-Hill, 2003

[7] G. Sindoni, L. Tininini - *"Statistical Warehousing on the Web: Navigating Troubled Waters"*, IEEE Computer Society - Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, 2006

[8] P. Atzeni, S. Ceri, P. Fraternali, S. Paraboschi, R. Torlone - *"Basi di Dati"* 2°ed., McGraw-Hill, 1999

[9] A. Vincenzi - *www.olap.it*, LAST VIEW: Maggio 2007

[10] M. Golfarelli, S. Rizzi - *"Data Warehouse – Teoria e Pratica della Progettazione"* 2°Ed., McGraw-Hill, 2006

[11] Dialog Sistemi – Information Technology Consulting, *www.dialog.it*, LAST VIEW: Maggio 2008

[12] Wikipedia, *it.wikipedia.org*, LAST VIEW: Maggio 2008

[13] L. Cabibbo - *"Il Modello Dimensionale",* Documento, Università Degli Studi Roma Tre, 2000

[14] L. Cabibbo, R. Torlone - *"Un Quadro Metodologico per la Costruzione e l'uso di un Data Warehouse"*, Convegno Nazionale Sistemi Evoluti per Basi di Dati, Ancona, pp. 123-140, Giugno 1998

[15] S. Kuhlins, R. Tredwell - *"Toolkits for Generating Wrappers"*, Revised Papers From The International Conference Netobjectdays On Objects, Components, Architectures, Services, and Applications for a Networked World, LNCS 2591, Springer-Verlag, pp. 184-198, 2002

[16] Massachusetts Institute of Technology, *groups.csail.mit.edu/uid/lapis*, LAST VIEW: Giugno 2008

[17] R. Miller, B. Myers - *"Lightweight Structured Text Processing"*, Proceedings of USENIX 1999 Annual Technical Conference, Monterey, CA, pp. 131-144, Giugno 1999

[18] R. Miller, B. Myers - *"Integrating a Command Shell into a Web Browser"*, Proceedings of USENIX 2000 Annual Technical Conference, San Diego, CA, pp. 171-182, Giugno 2000

[19] Tcl Developer Site, *www.tcl.tk,* LAST VIEW: Aprile 2008

[20] Sun Microsystems, *java.sun.com,* LAST VIEW: Aprile 2008

[21] Deitel & Deitel - *"JAVA – Fondamenti di Programmazione"*, Apogeo, 2000

[22] W3C, *www.w3.org/xml,* LAST VIEW: Aprile 2008

[23] M. Carducci - *"XML Pocket"*, Apogeo, 2005

[24] Sun Microsystems, *www.mysql.com,* LAST VIEW: Aprile 2008

[25] Deitel & Deitel - *"JAVA – Tecniche Avanzate di Programmazione"*, Apogeo, 2001

[26] T. Flagella - *"Jdbc per l'accesso Java a DB"*, Documento, *www.link.it*, LAST VIEW: Aprile 2008

[27] Pentaho Analysis Services, *mondrian.pentaho.org*, LAST VIEW: Giugno 2008

[28] Miriade Technology - *"Mondrian-JPivot, Business Intelligence con Strumenti Software Open Source", www.miriade.it*, LAST VIEW: Giugno 2006

[29] Apache Software Foundation, *tomcat.apache.org*, LAST VIEW: Aprile 2008

[30] S. Li, V. Chopra, B. Galbraith - *"Apache Tomcat - Guida per lo Sviluppatore"*, Hoepli, 2002

[31] G. Spofford, S. Harinath, C. Webb - *"MDX Solutions"*, Wiley, 2006

[32] M. Bonacorsi - *"BART: Uno Strumento di Analisi di Business e Reportistica"*, Università Degli Studi Di Modena e Reggio Emilia, 2006

[33] Miriade Technology - *"JasperReports & Eclipse BIRT, Soluzioni Open Source per Enterprise Reporting", www.miriade.it*, LAST VIEW: Giugno 2006

[34] Pentaho Analysis Services – Mondrian Developer Guide, *mondrian.pentaho.org/documentation*, LAST VIEW: Giugno 2008